A Level Computer Science

Exam Style Questions

Unit 1.4.2

Data Structures

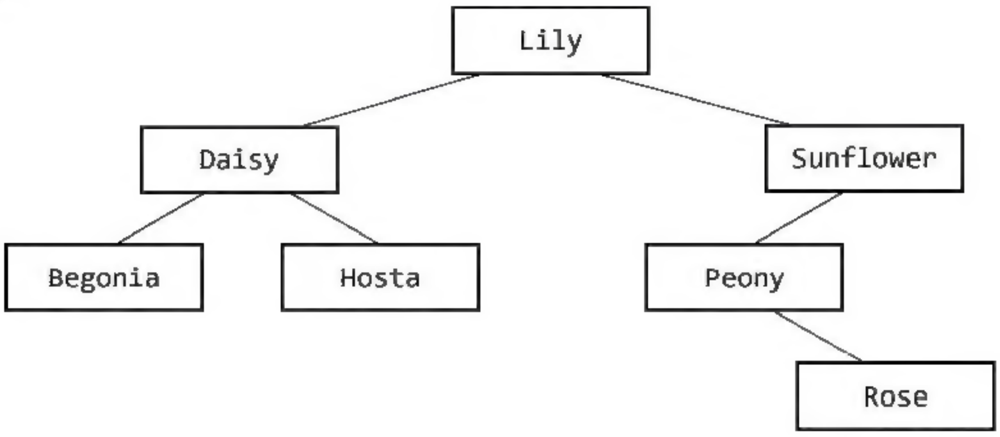
Linked Lists

Name		**Date**	

Score	Percentage	Grade
/ 31		

Question 1

A program needs to store the names of plants that are in a garden, so they can be easily found and accessed in alphabetical order. The data is stored in a tree structure. Part of the tree is shown.



The elements in the tree in above are read into a linked list producing an alphabetised list.

a) Complete the following table to show the linked list for the data.

Data Item	Data	NextPointer
0	Begonia	
1	Daisy	
2	Hosta	
3	Lily	
4	Peony	
5	Rose	
6	Sunflower	

[2]

A new plant, Lavender, needs adding to the linked list. The linked list needs to retain its alphabetical order.

b) Complete the table to show the linked list after Lavender is added.

Data Item	Data	NextPointer
0	Begonia	
1	Daisy	
2	Hosta	
3	Lily	
4	Peony	
5	Rose	
6	Sunflower	

[3]

c) Hosta needs removing from the linked list.

Explain how a data item is removed from a linked list. Use the removal of Hosta in your answer.

[4]

d) The linked list is stored as a 2D array with the identifier `plantList`. The index of the first element of the linked list is stored in the identifier `firstElement`.

All contents of the linked list need to be output in alphabetical order.

Write an algorithm to follow the pointers to output the contents of the linked list in alphabetical order.

Add comments to explain your code. Describe why a queue is a suitable structure for this program.

[5]

Question 2

A programmer is developing an ordering system for a fast food restaurant. When a member of staff inputs an order, it is added to a linked list for completion by the chefs.

a) Explain why a linked list is being used for the ordering system.

[2]

b) Each element in a linked list has:

- a pointer, `nodeNo`, which gives the number of that node
- the order number, `orderNo`
- a pointer, `next`, that points to the next node in the list

The table below shows the current contents of the linked list, orders.

nodeNo	orderNo	next
0	154	1
1	157	2
2	155	3
3	156	NULL

i. Order 158 has been made, and needs adding to the end of the linked list.

Add the order, 158, to the linked list as shown in the table above. Show the contents of the linked list in the following table.

nodeNo	orderNo	next

[2]

ii. Order 159 has been made. This order has a high priority and needs to be the second order in the linked list.

Add the order, 159, to the *original* linked list. Show the contents of the linked list in the following table.

nodeNo	orderNo	next

[3]

a) The linked list is implemented using a 2D array, theOrders:

- Row 0 stores orderNo
- Row 1 stores next

The data now stored in theOrders is shown in the table below.

184	186	185	187
1	2	3	

theOrders [1,0] would return 1

The following algorithm is written:

```
procedure x()

    finished = false
    count = 0
    while NOT(finished)
        if theOrders[1,count] == null then
            finished = true
        else
            output = theOrders[0, count]
            print(output)
            count = theOrders[1, count]
        endif
    endwhile

    output = theOrders[0, count]
  print(output)

endprocedure
```

i. Outline why nodeNo does not need to be stored in the array.

[1]

ii. Complete the trace table for procedure x, for the data shown in the table above.

finished	count	output

[3]

i. Describe the purpose of procedure `x()`

[2]

ii. A new order, 190, is to be added to `theOrders`. It needs to be the third element in the list. The current contents of the array are repeated here for reference:

184	186	185	187			
1	2	3				

Describe how the new order, 190, can be added to the array, so the linked list is read in the correct order, without rearranging the array elements.

[4]